What is GitHub?

GitHub is a version control and data storage solution used by programmers and data scientists. It can be useful to learn some basic GitHub concepts and commands to collect someone else's data or publish your data for others to use.

We're going to learn how to set up GitHub on our machines and interact with GitHub via the terminal.

Our Course Repository

You can find the link to our course repository here: https://github.com/bennoble/python_summer2021.

Git from Terminal

To clone/fork a repository:

1. Go to our class repository on GitHub (i.e., click the above link).
2. Click on `fork` in the top right corner. This will launch an animiation, after which, you'll end up on *your* forked repository. The url should read https://github.com//python_summer2021.
3. Click the green "code" button and click the clipboard icon next to the url.
4. Open terminal and use `cd` to go to folder where you want to clone the repository.
5. Type `git clone ...` where the ... is the link you copied from the website and then hit enter.

Now you need to let GitHub know that *my* repository is the origin repository so you can pull down changes later.

1. `cd` into the folder you just created. In terminal, type `git remote -v` and you will likely see:

   origin https://github.com/YOUR_USERNAME/YOUR_FORK.git (fetch)

   origin https://github.com/YOUR_USERNAME/YOUR_FORK.git (push)

2. Use the link from earlier that you copied from the website (aka my repository link) to tell GitHub my folder is the origin with `git remote add upstream ...`

3. If you did this correctly, when you type `git remote -v`, you should see something like:

   origin https://github.com/YOUR_USERNAME/YOUR_FORK.git (fetch)

   origin https://github.com/YOUR_USERNAME/YOUR_FORK.git (push)

   upstream https://github.com/ORIGINAL_OWNER/ORIGINAL_REPOSITORY.git (fetch)

   upstream https://github.com/ORIGINAL_OWNER/ORIGINAL_REPOSITORY.git (push)

Once you have this folder on your computer, you can work on files within it or save new files to it. However, any changes you make locally will not sync to GitHub until you `push` them.

To Push:

1. Use `cd` to navigate to the folder on your computer.
2. `git status` allows you to check your uncommitted changes.
3. `git add --all` adds all of your changes to be committed, `git add ...` and the filename adds just one file.
4. `git commit -m "..."` where you put an informative message in the ... to denote the changes/updates.

5. `git push` adds your changes to GitHub for all to see.

To Pull:

Pulling will add changes from the origin (i.e. my repository, when I post them) to your repository. It will not overwrite anything you've added on your end. Type `git pull upstream master`. It might ask you for a commit message—you can just leave this blank.

Bonus: Shortcuts!

All of these commands, so much typing. Yuck! It turns out that we can create our own shortcuts in terminal.

To create shortcuts (aka aliases), you need to access your bash profile with `touch ~/.bash_profile; open ~/.bash_profile`. This will open up of document with a bunch of stuff in it that you don't need to worry about. Go to the end of the existing text and create a new section called "Aliases."

To create a shortcut use the following syntax: `alias shortcut = "command"` where "command" is the actual command you'd use and shortcut is what you want the shortcut code to be.

For example, accessing the profile is a lot of code to remember, so I've created a shortcut to access it with the code: `alias profile="touch ~/.bash_profile; open ~/.bash_profile"`. Now, any time I open terminal and type `profile`, my computer will open this profile file.

You can create any kind of shortcut you want, so for example, to create a shortcut for `git pull upstream/master`, I could type: `alias pu="git pull upstream/master"` where "pu" stands for pull (but you could use anything here, like "q" or "kds", not just "pu", but of course you want to be able to easily remember it).